Presentationalcomponents

Overview

Presentational components are the atomic components in the PlayPlus architecture that focus purely on rendering and user interaction. They receive data through inputs and emit events through outputs, making them highly reusable and testable.

Key Characteristics

Pure Components

- Focus solely on presentation and user interaction
- No business logic or data fetching
- Receive all data through inputs
- Emit events through outputs

Unidirectional Data Flow

- Data flows in through inputs
- Events flow out through outputs
- No direct service dependencies
- Predictable and testable behavior

Accessibility First

- Built-in ARIA support
- Keyboard navigation
- Screen reader friendly
- Semantic HTML structure

Template Structure

TypeScript Component (component.ts.hbs)

HTML Template (component.html.hbs)

Features

Built-in Features

- Signal Inputs/Outputs Modern Angular 17+ signal-based inputs Type-safe data flow Reactive updates
- Accessibility ARIA labels and roles Keyboard navigation support Screen reader friendly Semantic HTML structure
- Style Customization Custom styles input Style sanitization for security CSS custom properties support
- Interaction States Disabled state handling Interactive state computation Event handling with proper validation

Best Practices

- Input Validation readonly data = input . required < ExampleData > ( ) ; readonly disabled = input < boolean > (false) ;
- Computed Values readonly isInteractive = computed (( ) => ! this . disabled ( ) && this . data ( ) . isActive) ;
- Event Handling protected onClick ( ) : void { if (this . isInteractive ( )) { this . itemClicked . emit (this . data ( )) ; } }
- Accessibility readonly ariaLabel = computed (( ) => ` ${ this . data ( ) . title } - ${ this . data ( ) . description || 'No description' } `) ;

Usage Examples

Basic Presentational Component

This creates:

- UserCardComponent with data input
- Action and click outputs
- Accessibility features
- Storybook stories

Advanced Presentational Component

This creates:

- ProductTileComponent with Product interface
- Custom styling support
- Comprehensive interaction handling

Integration with Container Components

Presentational components are used by container components:

Data Interface Pattern

Each presentational component defines its own data interface:

This ensures:

- Type safety
- Clear contract between components
- Easy refactoring
- Better IDE support

Styling and Theming

CSS Custom Properties

Custom Styles Input

Testing

Presentational components include comprehensive tests:

Architecture Benefits

- Reusability : Can be used across different containers
- Testability : Pure functions are easy to test
- Maintainability : Clear separation of concerns
- Performance : OnPush change detection
- Accessibility : Built-in a11y features

Common Patterns

List Items

Cards

Forms

Next Steps

After creating a presentational component:

- Define the data interface for your specific use case
- Customize the template with your content
- Add styling to match your design system
- Create stories for visual testing
- Write tests for all interaction scenarios
- Use in container components for data integration

Developer Checklist

Before Creating Presentational Components:

- Is the component purely presentational (no business logic)?
- Are all inputs typed with TypeScript interfaces?
- Are all user interactions handled through outputs?
- Is OnPush change detection enabled?
- Are computed values used for derived state?
- Is disabled state properly handled?
- Are all interactive elements keyboard accessible?
- Do I have unit tests for all interaction scenarios?
- Is the component reusable across contexts?
- Are style customization options provided?